OPTIMIZING AN UNKNOWN FUNCTION BY THE METHOD OF BOUNDED LEAST SQUARES

RALPH V. BUCK

OPTIMIZING AN UNKNOWN FUNCTION

BY THE METHOD OF

BOUNDED LEAST SQUARES

by

Ralph V. Buck

Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE
IN
OPERATIONS RESEARCH

United States Naval Postgraduate School
Monterey, California

1 9 6 5

OPTIMIZING AN UNKNOWN FUNCTION
BY THE METHOD OF
BOUNDED LEAST SQUARES

by

Ralph V. Buck

This work is accepted as fulfilling

the thesis requirements for the degree of

MASTER OF SCIENCE

IN

OPERATIONS RESEARCH

from the

United States Naval Postgraduate School

ABSTRACT

The problem of estimating the position of an extreme point of an unknown function of several independent variables is examined for the case where the dependent variable is known to be bounded. The classical method of least sequres is formulated as a quadratic programming problem to be solved numerically on a digital computer, where the coefficients of the fitted equation are determined subject to restrictions on both the independent variables and the dependent variable.

Several two dimensional models were examined using synthetic experimental design techniques. The results, though not conclusive, indicate that the method of bounded least squares can be a useful computational tool in some two dimensional problems. It remains to be shown whether the algorithm is useful in problems involving more than two independent variables.

ACKNOWLEDGEMENTS

Special thanks are due to Professor H. J. Larson for his great patience, advice, and technical supervision during the preparation of this thesis.

The author is also indebted to Professor R. H. Shudde for his penetrating suggestions during the construction of the computer code.

Finally, the author greatly appreciates the fine cooperation extended by the Computer Center of the U. S. Naval Postgraduate School.

R. V. B.

U. S. Naval Postgraduate School
Monterey, California
May, 1965

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOTATION

SYMBOL	PAGE	MEANING
a, b_i, c_{ij}	2	coefficients of the x_i.
d_i	2	upper bound on x_i.
e	8	lower bound on Y.
f	8	upper bound on Y.
g_i	2	lower bound on x_i.
G	7	residual.
H	4	unknown true response function.
kn	8	number of coefficients in the fitted equation.
m	7	number of experiments.
n	2	number of independent variables.
R	2	experimental region in n-space.
r	11	distance between true and estimated extreme point.
$\bar{r}$	11	average extreme point error.
X_i	19	the independent variables in coded form.
x_i	2	the independent variables.
x_i^*	11	the value of x_i at a true extreme point.
x_i^{**}	4	the value of x_i at an estimated extreme point.
Y	2	calculated value of the response.
Y'	7	experimental value of the response.
Y^*	4	value of the response at an extreme point.

INTRODUCTION

The method of least squares to fit a curve or surface to a set of data points is well known in the literature, as are techniques for determining the extreme points of such a surface. In particular, non-linear methods are available [10] to examine the surface in a particular region defined by bounds on the independent variables.

While considering the special problem of approximating, by a quadratic equation, a complex probability distribution, it was proposed to make use of optimizing techniques which would give a good estimate of the particular combination of independent variables producing maximum probability. Since the equation was only an approximation, it was expected that errors would occur in the estimation and that further accuracy would be obtained by testing in the physical system.

At this point it occured to the author that a large class of problems of this type must exist; i.e., certain independent variables are known to be bounded, and it is desired to find an extreme point of some function of these variables, which is itself bounded.

A considerable body of literature on the theoretical nature of non-linear programming exists to consider the bounded independent variables [7] [11], and it seemed reasonable to extend these methods to treat the bounded dependent variable in such a manner that the extreme point of the true Response Surface could be more closely estimated.

STATEMENT OF THE PROBLEM

Consider a physical process or system under investigation which has at least one characteristic called the Response, such as cost, yield, power, etc. Assume that this Response is some unknown function, called the Response Function, of a set of independent variables and that these variables can each be quantitatively fixed at a given value, or level. The observed response resulting from a combination of the variables, however, is the sum of the true Response and a random error which is distributed about a mean of zero with constant variance (no matter what the values of the system variables).

Geometrically, the Response Function may be represented by a surface in (n+1)-space called the Response Surface. The experimenter has systematically proceeded to a particular region, R, in n-space (defined by the n independent variables) which is bounded by the planes $x_i = d_i$ and $x_i = g_i$ $(i=1,2,..,n)$. He then wishes to find the levels of the independent variables for which the Response is extreme, restricting the independent variables to lie within R.

To do this he must make the assumption that the Response Function may be adequately represented by some analytic function. In our case we shall assume that a quadratic function of the form

$$Y = a + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j \tag{1}$$

will adequately represent the Response Function, at least in the vicinity of an extreme point.

If we also assume that the experimenter has some prior knowledge that the Response Function is bounded, then he should be able to make use of this knowledge in any procedure which estimates the coefficients of the quadratic equation used to locate the extreme point.

CLASSICAL METHOD OF SOLUTION

In a classical solution to a problem of this type, experiments would be performed at certain selected points in the region, R, and the method of least squares [9] would then be employed to determine the coefficients of the equation to be fitted.

Once the quadratic has been estimated, the set of partial differential equations derived by differentiating the function, Y, with respect to each x_i can be solved for the location of the extreme points. In order to remain within the region defined by the bounded independent variables, nonlinear programming methods may be applied to determine the position of the bounded extrema [10].

We notice that no restriction is placed on the Response, Y. The optimizing technique has given the set $\{x_i^{**};\ i=1,..,n\}$ which corresponds to an extreme point, $Y^{*}=H(x_1^{**},...,x_n^{**})$, of the fitted equation only. It is quite possible that Y^{*} does not lie within the a priori bounds on the true Response and, in practice, further experimentation could be performed at and near $\{x_1^{**},...,x_n^{**}\}$ in order to confirm the existence of the extreme point and to more exactly determine the Response at that point. If the research worker has the time and resources available to conduct the large number of experiments necessary to fit the quadratic by the method of least squares and then to converge on the true position of extreme response from a predicted position, such a method may be entirely adequate.

It was felt, however, that such a procedure was not as efficient in predicting an extreme point as should be possible when a priori bounds on the Response are known. Accordingly, the classical method was examined to determine where a modification to the theory could best be made.

THE METHOD OF BOUNDED LEAST SQUARES

Though further experimentation is to be performed in the vicinity of the predicted extreme response, the problem of optimization is concerned not only with predicting response, but with predicting the $Y^*=H(x_1^{**},\ldots,x_n^{**})$ with sufficient precision that the least number of experiments are necessary to converge to an extreme point.

In the classical solution, two options are available. First, the bounds on the Response may be ignored and the position of maximum response obtained from the fitted equation may be taken as the best estimate of the values of the independent variables which will maximize (or minimize) the observed response. Secondly, the hyperplanes representing the upper and/or lower bounds on the Response may be intersected with the Response Surface obtained by the estimate of the quadratic equation. Only those portions of the fitted response surface lying within the bounded region would be searched for an extreme point (see figure 1). This method was not considered to yield an acceptable solution since the entire range of values of the independent variables corresponding to the portion of the fitted response surface outside the bounded region would give an extreme, restricted response.

Situations can exist in which the fitted response surface varies greatly from the true Response Surface, both in shape and range of value over the surface. Thus, it was hypothesized that the extreme point of the fitted response surface could be moved

closer in n-space to the extreme point of the true Response Surface if the quadratic equation used to represent the true Surface could be estimated subject to the restriction that no point on the fitted surface could fall outside the a priori bounds on the true Response (see figure 2).

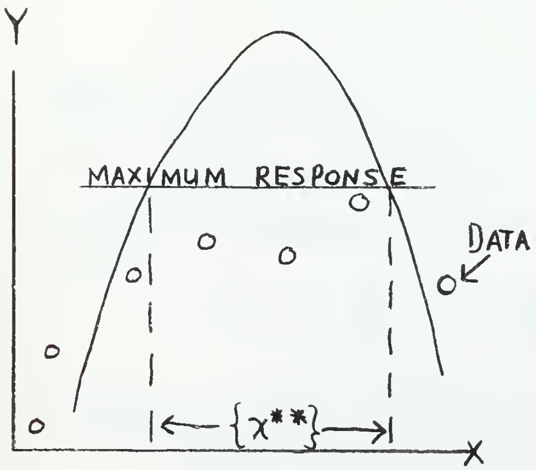

FIGURE 1

TRUNCATED LEAST SQUARES FIT

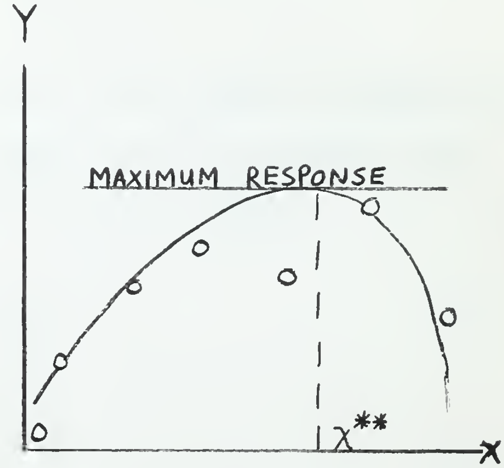

FIGURE 2

BOUNDED LEAST SQUARES FIT

Recall that equation (1) defined the fitted, or calculated response. If Y' is defined as the observed response, then $\left\{Y_k', Y_k\right\}$ $(k=1,\ldots,m)$ are the sets of response values obtained when m different combinations of values of the x_i are 1) set as levels in experiments where Y' is observed and 2) used to calculate Y from the fitted equation. Then,

$$G = \sum_{k=1}^{m} (Y_k - Y_k')^2 \tag{2}$$

is the sum of squares of the residuals. The sum, G, is sometimes referred to simply as the Residual.

Thus, the classical least squares method of estimating the coefficients may be modified as follows:

$$\text{Minimize } G \text{ with respect to all coefficients of the } x_i \tag{3}$$

$$\text{subject to } g_i \leq x_i \leq d_i \ (i=1,\ldots,n) \tag{4}$$

$$\text{and } e \leq Y_k \leq f \ (k=1,\ldots,m) \tag{5}$$

where n is the number of variables and m is the number of experiments (data points). In Appendix A it is shown that an equivalent form of (3) is to Minimize F_1 with respect to all A_i, where

$$F_1 = Z_s + \sum_{i=1}^{kn} C_i A_i + \sum_{j=1}^{kn} CK_i A_i^{\,2} \tag{6}$$

and that (5) may be restated as

$$F_j \geq 0 \ (j=2,\ldots,2m+1) \tag{7}$$

Non-linear programming methods designed to accept problem statements in the above form, where the constraints are linear in the unknowns, are classed as quadratic programming. The particular algorithm used in this paper is due to Klingman[1] [8], and was selected for its general application to the case where non-linear constraints are encountered in addition to bounded Response, as well as its availability in finished program form.

[1]The program originally published in Klingman's thesis was modified after private communications with W. R. Klingman and Dr. D. M. Himmelblau, University of Texas, Austin, Texas.

EXPERIMENTAL PROCEDURE

Almost any convenient and reasonable method may be used to select the points at which the experiments are to be performed. In keeping with the image of an economy-minded experimenter, however, it was decided to make use of the techniques of composite design due to Box [1] [4]. As shown in Table I, a considerable reduction in the number of points to investigate is possible when the 2^n factorial design [5] is augmented rather than using the 3^n factorial design usually employed when quadratic effects are being observed.

TABLE I

COMPARISON OF COMPOSITE DESIGN
AND FACTORIAL DESIGNS

no. of variables	no. of experiments required		
	composite design	3^n design	2^n design
2	9	9	4
3	15	27	8
4	25	81	16
5	43	243	32
6	77	729	64
7	143	2187	128
8	273	6561	256

In order to simulate actual experimentation, a hypothetical physical system was represented by an analytic equation. For each combination of independent variables selected by the design program in Appendix B, the Response was evaluated and a random error distributed with zero mean and constant variance was added to it to obtain Y'_k $(k=1,\ldots,m)$. Using the data points thus generated, the

coefficients of the quadratic equation were estimated by 1) the classical method of least squares [6] [9] and 2) the method of bounded least squares (Appendix B). This procedure was repeated until a total of ten estimates of the equation were obtained for a constant response error variance, for each of four analytic equations tested, the independent variables being set at the same levels for both methods. Each of the fitted equations was then solved for an extreme point in the design region. The distance, in n-space, between the true extreme point and the extreme point of each fitted equation was determined. Finally, for each case, consisting of ten runs for a particular equation, the distances were averaged to obtain the average extreme point error, $\bar{r}$.

RESULTS

For each case, the distance in n-space between the true and estimated position of extreme Response was found using the relation

$$r = \sqrt{\sum_{i=1}^{n} (x_i^{**} - x_i^{*})^2} \tag{8}$$

and then averaging over the number of runs to obtain $\bar{r}$. This procedure was followed for the results of the classical least squares fit and the bounded least squares fit for each case.

Numerical Results

CASE ONE : $Y = 4x_1x_2 - 2x_1^2 - x_2^4$

The response error variance was set at .25 and the region, R, was defined by $\{(x_1, x_2): 0 \le x_1 \le 2,\ 0 \le x_2 \le 2\}$. A maximum is located at (1,1) and an a priori upper bound on the Response was assumed to be 1.0 . Using the method of bounded least squares, the estimated maximum was located 12.7 percent closer to the true maximum than by the classical method.

CASE TWO : $Y = -x^4 + 2x^3 + 2x^2 - 2x + 10$

The response error variance was set at 1.0 and the region, R, was defined by $\{x: -2 \le x \le 3\}$. An a priori upper bound on the Response was assumed to be 15.0 and two maxima are in R; a local maximum at $x = -1.0$ and a global maximum at $x = 2.0$. Using the bounded least squares method, the estimate of the extreme point was only 1.2 percent closer to the true maximum than by using the classical method. This difference was found to be completely submerged in the effect

of response error variance upon the location of the maximum.

TABLE II

COMPARISON OF MEAN EXTREME
POINT ERROR FOR CASE ONE

RUN	Classical Method			Bounded Method		
	x_1^{**}	x_2^{**}	r	x_1^{**}	x_2^{**}	r
10	0.7007	0.6103	0.4914	0.7188	0.6403	0.4566
9	.7614	.6518	.4219	.5721	.5986	.5867
8	.7607	.6748	.4037	.6647	.6228	.5047
7	.6593	.5934	.5304	.5177	.5758	.6423
6	.6403	.6133	.5282	.6961	.6568	.4585
5	.7932	.6993	.3647	.6312	.5483	.5832
4	.7650	.6494	.4219	.6181	.6277	.5334
3	.6004	.5493	.6022	.7293	.6308	.4578
2	.0000	.0000	1.4142	.5742	.5389	.6277
1	.2386	.4357	.9476	.7303	.5787	.4970
$\bar{r}$			.6126			.5348

CASE THREE : $Y = \dfrac{1}{(x_1 + 1)^2 + x_2^2 + 10^{-8}}$

The response error variance was set at .0001 and R was defined by $\left\{(x_1, x_2) : -4 \leq x_1 \leq 4, -4 \leq x_2 \leq 4\right\}$. The Response Surface is symmetrically oriented about a maximum located at (-1,0), and an a priori upper bound of 1.0 was placed on the Response. Both methods of fitting the experimental points resulted in identical estimates of the maximum at (0,0) for all runs. It should be noted that the Response Surface is very shallow outside that area enclosed

by a cylinder of radius 1.0 which is centered at (-1,0).

TABLE III

COMPARISON OF MEAN EXTREME
POINT ERROR FOR CASE TWO

RUN	Classical Method x_1^{**}	Classical Method r	Bounded Method x_1^{**}	Bounded Method r
10	0.5669	1.4331	0.5706	1.4294
9	.5603	1.4397	.5670	1.4330
8	.5488	1.4512	.5853	1.4147
7	.5603	1.4397	.5623	1.4377
6	.5636	1.4364	.5897	1.4103
5	.5431	1.4569	.5670	1.4330
4	.5540	1.4460	.5735	1.4365
3	.5400	1.4600	.5809	1.4191
2	.5702	1.4298	.5537	1.4463
1	.5636	1.4364	.5980	1.4020
$\bar{r}$		1.4429		1.4251

CASE FOUR : $Y = x_1^2 - x_2^3 - 3x_1x_2$

The response error variance was set at .01 and an a priori lower bound of -1.69 was placed upon the Response. The region, R, was defined by $\left\{(x_1,x_2): -1 \leq x_1 \leq 3, -1 \leq x_2 \leq 2\right\}$. There is a saddle point at (0,0), a minimum at (2.25,1.5), and the global minimum for R is located at (-1,-1). Both methods correctly estimated the position of the minimum for all runs.

Computational Results

Program DESIGN required about one minute to compile and execute for two variables, including printout and punch. Program FIT required approximately three minutes to compile and four seconds to execute for each run. The minimum fractional change in both the coefficients being estimated and the value of the Residual being tested was set at 10^{-7}. In his thesis, Klingman [8] gives a compile time of about one and one-half minutes for the quadratic program (subroutines DIRECT, PATSER, EXPSER, GRAD, SUMS and OUTPUT in program FIT of this paper).

The program due to Dillon [6] was used to fit the quadratic by the classical method. That program consists of the elements of programs DESIGN and FIT with the equations from Krase and Cyl-Champlin [9] substituted for the subroutine package which comprises the quadratic program portion of FIT. The program required one and one-half minutes to compile and less than one second to execute for each run. Thus, the classical method of solution required about one-half the total computer time required by the bounded least squares method. As mentioned before, no attempt was made to reduce either compile or execution time, for either program, to a minimum.

CONCLUSIONS

The algorithm presented here was expected to provide a closer estimate, in terms of the physical values of the independent variables, of the location of a point of extreme Response. It was assumed that the method would be of greatest value in cases where the Response Surface 1) is non-symmetrical about an extreme point in the interior of the region being investigated or 2) has more than one extreme point in the same region.

In no case tested did the algorithm provide worse estimates of the true extreme point than did the classical method. For cases three and four, where the Surfaces were symmetric, and oriented with an extreme point at the boundary, no particular advantage accrued from using the method of bounded least squares. In case two the results were inconclusive, and case one did not produce the desired improvement over the classical method, even though a distinct improvement was observed.

For one and two dimensional problems, it is felt that the advantage of the system is outweighted by the disadvantage of observing the Response at only a small number of points. Any method involving least squares requires enough points to adequately represent the Surface being approximated. In the bounded least squares method, where the advantage lies in estimating an extreme point of a complex surface, composite design point selection does not choose enough points to adequately define the surface. Further

tests should be made to determine the minimum number of points necessary to permit the method of bounded least squares to enjoy a distinct advantage over classical methods.

No suitable problems of dimensionality three or higher were found which permitted rigorous testing of this algorithm. Since the number of experimental points required for these problems becomes quite large (see Table I), a significant advantage may be realized by the method of bounded least squares, since additional information is included in the solution to the problem at no experimental cost. The cost of further testing, as opposed to doubling the computer time, must be considered for each system investigated, before the decision is made to select any method which reduces the number of experiments.

BIBLIOGRAPHY

1. Box, G. E. P., "The Exploration and Exploitation of Response Surfaces: Some General Considerations and Examples, "Biometrics, Vol. 10, 1954, pp16-60.

2. ___, "Some General Considerations in Process Optimisation," Technical Report No. 13, Princeton University, Princeton, New Jersey, [n.d.] .

3. ___ and J. S. Hunter, "Multi-Factor Experimental Designs for Exploring Response Surfaces," Annals of Mathematical Statistics, Vol. 28, 1957, pp195-241.

4. ___ ___ K.B. Wilson, "On the Experimental Attainment of Optimum Conditions," Journal of the Royal Statistical Society, Series B, Vol. XIII No. 1, 1951, pp1-45.

5. Davies, O. L. (editor), The Design and Analysis of Industrial Experiments, Hafner and Company, New York, 1956.

6. Dillon, J. C., "Synthetic Experimental Design and Optimization." Unpublished AI-memo 4955, Atomics International division of North American Aviation, Inc., Canoga Park, California, 1960.

7. Dorn, W. S., "Non-Linear Programming - A Survey," Management Science, Vol. 9, 1963, pp171-208.

8. Klingman, W. R., "Non-linear Programming by the Multiple Gradient Summation Technique." Unpublished Master's thesis, The University of Texas, Austin, Texas, 1963.

9. Krase, J. M. and C. Cyl-Champlin, "Synthetic Experiment Design Techniques in Reactor Analysis." GE-ANPD, APEX 303 (Nov., 1956).

10. Selengut, D. S., "Optimizing Nuclear Designs by Means of Non-linear Programming," GE-ANPD, APEX 378 (Oct., 1957).

11. Tucker, A. W., "Linear and Non-Linear Programming," Operations Research, Vol. 5, 1957, pp244-257.

APPENDICES

APPENDIX A

FORMULATION OF THE METHOD OF BOUNDED LEAST SQUARES AS A QUADRATIC PROGRAMMING PROBLEM

All terms used in this Appendix have been previously defined except the following:

δ : magnitude of the bounds on the coded values of all the independent variables, X_i.

K : constant due to orthogonality requirement which permits independent evaluation of each coefficient.

B_o, B_i, $(BB)_{ij}$: coefficients of the orthogonal quadratic equation, for coded X_i.

(NOTE : double letter symbols enclosed by ( ) refer to names of single variables.)

The form of the quadratic equation with which we desire to fit the experimental data is

$$Y = B_0 + \sum_{i=1}^{n} B_i X_i + \sum_{i=1}^{n} (BB)_{ii}(X_i^2 - K) + \sum_{i=1}^{n} \sum_{j>i}^{n} (BB)_{ij} X_i X_j \tag{A-1}$$

so that the sum of squares of the residuals may be writted as

$$G = \sum_{k=1}^{n} (Y_k' - Y_k)^2 = \sum_{k=1}^{m} (Y_k')^2 + \sum_{k=1}^{m} Y_k^2 - 2 \sum_{k=1}^{m} Y_k' Y_k \tag{A-2}$$

Simplifying each term,

$$\sum_{k=1}^{m} (Y_k')^2 = \text{constant} = Z_s \tag{A-3}$$

and

$$\sum_{k=1}^{m} Y_k^2 = \sum_{k=1}^{m} B_o^2 + \sum_{k=1}^{m} \sum_{i=1}^{n} (B_i X_{ik})^2$$

$$+ \sum_{k=1}^{m} \sum_{i=1}^{n} (BB)_{ii} (X_{ik}^2 - K)^2$$

$$+ \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j>i}^{n} ((BB)_{ij} X_{ik} X_{jk})^2 \qquad \text{(A-4)}$$

where all other terms can be shown to be zero. Then, using the relations developed by Krase [9] for classical least squares theory using orthogonal designs,

$$K = \frac{1}{m} \sum_{k=1}^{m} X_{ik}^2 \quad \text{for any } i=1,2,\ldots,n \qquad \text{(A-5a)}$$

$$= \frac{1}{m}(2^n - 2\delta^2) = \sqrt{2^n/m} \qquad \text{(A-5b)}$$

$$\delta^2 = \frac{1}{2}(\sqrt{2^n/m} - 2^n) \qquad \text{(A-5c)}$$

$$\sum_{k=1}^{m} (X_{ik}^2 - K)^2 = 2\delta^4 \quad \text{for all } i=1,\ldots,n \qquad \text{(A-5d)}$$

$$\sum_{k=1}^{m} X_{ik}^2 X_{jk}^2 = 2^n \quad \text{for all } i \neq j \qquad \text{(A-5e)}$$

$$\sum_{k=1}^{m} (X_{ik}^2 - K) = 0 \quad \text{for all } i=1,\ldots,n \qquad \text{(A-5f)}$$

it is easily shown that

$$\sum_{k=1}^{m} Y_k^2 = mB_o^2 + mK \sum_{i=1}^{n} B_i^2$$

(continued) $$+ 2\delta^4 \sum_{i=1}^{n} (BB)_{ii}^2$$

$$+ 2^n \sum_{i=1}^{n} \sum_{j>i}^{n} (BB)_{ij}^2 \qquad \text{(A-6)}$$

The last term of (A-2) is a straightforward sum of multiplications. When terms are combined,

$$G = Z_s + mB_o^2 + mK \sum_{i=1}^{n} B_i^2 + 2\delta^4 \sum_{i=1}^{n} (BB)_{ii}^2$$

$$+ 2^n \sum_{i=1}^{n} \sum_{j=i}^{n} (BB)_{ij}^2 - 2 \sum_{k=1}^{m} Y_k' B_o$$

$$- \sum_{i=1}^{n} B_i \left\{2 \sum_{k=1}^{m} (X_{ik} Y_k')\right\}$$

$$- \sum_{i=1}^{n} (BB)_{ii} \left\{2 \sum_{k=1}^{m} Y_k'(X_{ik}^2 - K)\right\}$$

$$- \sum_{i=1}^{n} \sum_{j=i}^{n} (BB)_{ij} (2 \sum_{k=1}^{m} Y_k' X_{ik} X_{jk}) \qquad \text{(A-7)}$$

Then simplifying

$$G = Z_s + Z_o B_o + \sum_{i=1}^{n} Z_i B_i + \sum_{i=1}^{n} \sum_{j=1}^{n} (ZZ)_{ij} (BB)_{ij}$$

$$+ Q_o B_o^2 + \sum_{i=1}^{n} Q_i B_i^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} (QQ)_{ij} (BB)_{ij}^2 \qquad \text{(A-8)}$$

where the substitutions are obvious.

The Response is bounded by an upper and lower restriction which can respectively approach plus and minus infinity;

$$e \leq Y_k \leq f \qquad (k=1,2,\ldots,m)$$

Then, (A-9) can be restated as

$$\begin{aligned} Y_k - e &\geq 0 \\ f - Y_k &\geq 0 \end{aligned} \qquad (k=1,2,\ldots,m) \qquad \text{(A-10)}$$

or, using (A-1) and the transformation

$$Y_k = \sum_{i=1}^{kn} A_i (XX)_{ik} \tag{A-11}$$

a change of variables may be made so that

$$\begin{aligned} F_i &\geq 0 \qquad (i = 2, 3, \ldots, m+1) \\ F_j &\geq 0 \qquad (j = m+2, \ldots, 2m+1) \end{aligned} \tag{A-12}$$

Similarly, (A-8) may be transformed to

$$F_1 = G + Z_s + \sum_{i=1}^{kn} C_i A_i + \sum_{i=1}^{kn} (CK)_i A_i^2 \tag{A-13}$$

Equivalent quantities for the case where the number of variables, n, is equal to two are listed in Table IV. The equations (A-12) and (A-13), together with the bounds on the X_i, constitute the form of the quadratic programming problem solved in Appendix B.

TABLE IV

EQUIVALENT QUANTITIES IN THE QUADRATIC PROGRAMMING PROBLEM FOR TWO VARIABLES

Unknown Coefficients		Objective Constants			Constraint Constants (k = 1,2,...,m)	
B_0	A_1	Z_o	C_1	relations for Q, CK are similar to those for Z, C	1.0	$XX_{1,k}$
B_1	A_2	Z_1	C_2		$X_{1,k}$	$XX_{2,k}$
B_2	A_3	Z_2	C_3		$X_{2,k}$	$XX_{3,k}$
BB_{11}	A_4	ZZ_{11}	C_4		$X_{1,k}^2 - K$	$XX_{4,k}$
BB_{12}	A_5	ZZ_{12}	C_5		$X_{1,k}X_{2,k}$	$XX_{5,k}$
BB_{22}	A_6	ZZ_{22}	C_6		$X_{2,k}^2 - K$	$XX_{6,k}$

APPENDIX B

The FORTRAN code for selecting the experimental points (program DESIGN) and for finding the coefficients fo the quadratic equation by the method of bounded least squares (program FIT) is described here.[2] The production runs were made on a CDC 1604 digital computer under COOP Monitor control. The programs allow considerable generality as written, and no attempt was made to "clean up" either one for the specific use of a bounded least squares fit.

A general flow diagram is not considered essential to an understanding of either program DESIGN or FIT. Subroutines DIRECT, PATSER, EXPSER and SUMS comprise the Multiple Gradient Summation algorithm [8] used to solve the quadratic programming problem. Detailed flow charts for these routines are available from other sources.[3]

As presently programmed, a maximum of seven variables can be considered. On a 32K machine additional variables can be included if only one bound (upper or lower) is placed on the Response. Subroutines SUMS and GRAD must then be modified appropriately.

[2]The algorithms given in [6] and [8] were modified and combined for this use.

[3]Dr. D. M. Himmelblau, The University of Texas, Chemical Engineering Department, Austin, Texas.

1. FORTRAN Nomenclature.

Only the essential symbols used in the FORTRAN programs of this appendix are listed.

A(I)	-	The coefficients to be found.
AK	-	Orthogonality constant (K in Appendix A).
ALN(I)	-	The best set of coefficients found.
ALO(I)	-	Previous base value of A(I).
BZRO,B(I), BB(IJ)	-	Coded coefficients of the fitted response function.
BO	-	Coefficient (see REALB, REALC).
C(II)	-	Renumbered constants.
CINE(I)	-	Variable used to determine first negative constraint crossed.
CK(LJ)	-	Renumbered constants.
DEL(I)	-	Incremental move for each A(I).
DELT	-	Bound on the coded variables (δ in App. A).
EP(I)	-	Incremental change in A(I).
EV	-	Estimate of error variance of the fitted response values at the design points.
EX(I,J)	-	Physical value of independent variable I in experiment J.
F(I)	-	The objective function (Residual) and the constraints.
FCHG	-	Minimum allowable fractional change in F(1) before a search failure occurs.
FN	-	Value of F(1) tested in types I and II exploratory searches.
FNN	-	Value of F(1) calculated in subroutine SUMS.

FNEG(I)	-	The negative constraints when a non-feasible initial guess of A(I) is made.
G	-	Sum of squares of the residuals.
IACCEL	-	Accelerating factor in the adaptive move.
INK	-	Frequency for printing values during a successful pattern move sequence.
ILATE(I)	-	Latest constraint contacted during an adaptive move.
ITED	-	Search failure counter.
ITER	-	Pattern move failure counter.
K	-	Used only in DIRECT and associated subroutines as the number of unknowns (coefficients) to find.
KCON	-	Total number of constraints contacted in an adaptive move.
KDAPT	-	Counter for the number of adaptive moves.
KFREQ	-	Pattern move/exploratory search failure printout frequency.
KFREQS	-	Successful pattern move printout frequency.
KN	-	Number of coefficients of the fitted response function (also number of each type of coefficients C and CK).
LX,LY	-	If equal to 2, the number of search and pattern move failures/successes will be printed. If equal to 1, no printing will be done.
M	-	Number of experiments.
MK	-	Counter for exploratory search failures.
MM	-	Counter for successive exploratory search failures. Used to reduce step size.
MQ	-	Maximum number of allowable successful pattern moves.
N	-	Number of independent variables (only for main programs DESIGN and FIT and subroutine REAL)-maximum of seven.

		Number of constraints plus one (only for subroutines DIRECT, PATSER, EXPSER, SUMS, GRAD and OUTPUT) - equals 2M + 1.
NC	-	Identifying case number.
NOMEX	-	Counter for search failures.
NOPTN	-	Counter for successful pattern moves.
NUMEX	-	Counter for number of exploratory searches.
NX	-	Number of extra points at which to evaluate the fitted response function.
PARTIAL(I,J)	-	Partial derivatives of F(I) with respect to A(J).
PER(I)	-	DEL(I) equals PER(I)*ALO(I).
PERC	-	Initial fractional incremental move for each A(I).
QO,Q(I), QQ(J)	-	Constants as defined in Appendix A.
REALB(I), REALC(I,J)	-	Coefficients of the fitted response function when physical values of the independent variables are used. (see BO also)
RL, RU	-	Lower and upper bound on the Response.
SLOPE(I)	-	Factor used to convert to coded values of the independent variables.
SUD(I)	-	Number of successful successive search moves for each variable.
TEST	-	If PER(I) < TEST, the problem has converged to a final answer.
VECTOR(I)	-	Sum of gradients of the contacted constraints in an adaptive move.
WAY	-	Set equal to -1.0 to indicate that a minimum of the objective function is desired.
X(I,L)	-	Coded value of the independent variable I in experiment L.

XAV(I)	-	Constant used to convert to coded values of the independent variables.
XMAX(I)	-	Upper bound on the EX(I,J) for all J.
XMIN(I)	-	Lower bound on the EX(I,J) for all J.
XX(I,L)	-	Constants as defined in Appendix A.
Y(L)	-	Observed response (except in subroutine PATSER).
		Temporary storage in subroutine PATSER.
YD(L)	-	Residuals for each experiment.
YY(L)	-	Computed value of the response function.
ZO,Z(I), ZZ(J)	-	Constants as defined in Appendix A.

2. Input for Program DESIGN.

Card One	:	FORMAT (2I2) . . . . .N, NC
Card Two	:	FORMAT (6E12.8) . . . .XMIN(I), XMAX(I) punched in pairs for each variable, I = 1,N. Maximum of three pairs per card.

3. Input for Program FIT.

Card One	:	FORMAT (2I5,E12.8) . .N, M, DELT
Card Two	:	FORMAT (6E12.7) . . . .SLOPE(I), XAV(I) Punched in pairs, I = 1,N. May require up to three cards for this data.
Next Cards: (total of M cards)		FORMAT (8E10.3) . . . .X(I,L) up to eight values of the independent variables for the Lth experiment.
Next Cards:		FORMAT (8E10.6) . . . .Y(L) observed response for up to eight experiments per card.
Next Card	:	FORMAT (2I5,2E12.8) . .NC, NX, RU, RL
Next Card	:	FORMAT (80 character Hollerith title)
Next Card	:	FORMAT (I5,F14.8) . . .MQ, PERC
Next Card	:	FORMAT (8E10.6) . . . .Initial guesses for values of the A(I).
Next Card	:	FORMAT (E10.6,2I5). . .TEST, LX, LY
Next Card	:	FORMAT (E10.6,2I5). . .FCHG, KFREQS, KFREQ
Last Cards:		FORMAT (8E10.6) . . . .EX(I,NX) Extra points to evaluate. Punch I independent variable values per card; NX cards.

```
      PROGRAM DESIGN
      DIMENSION X(8,273), EX(8,273)
      COMMON/BLOCK2/SLOPE(8), XAV(8)
      COMMON/BLOCK3/XMIN(8), XMAX(8)
C
 1100 READ 101, N, NC, (XMIN(I),XMAX(I), I=1,N)
  101 FORMAT (2I2/(6E12.8))
      IF(N .EQ. 0)1101,1103
 1101 STOP
 1103 M= 1 + 2*N + 2**N
      DO 1102 I=1,N
 1102 XAV(I) = (XMIN(I) + XMAX(I))/2.
      DELT = SQRTF((SQRTF(M*2.**N) - 2.**N)/2.)
      N2 = 2**N
      DO 4 I=1,8    $  DO 4 L=1,N2
    4 X(I,L) = 0.
      DO 7 I=1,N   $  DO 7 L=1,N2
    7 X(I,L) = ((MODF(INTF((L/(2.**(I-1.))) + .99), 2.))*(-2.)) + 1.
      N3 = N2 + 1
      DO 10 I=1,8    $  DO 10 L=N3,M
   10 X(I,L) = 0.
      DO 14 I=1,N   $  DO 14 L=N3,M
      II = L - N2
      IF(I .EQ. II)11,12
   11 X(I,L) = -DELT
   12 III = II - N
      IF(I .EQ. III)13,14
   13 X(I,L) = DELT
   14 CONTINUE
      DO 1115 I=1,N
      SLOPE(I) = (XMAX(I) - XMIN(I))/(2.*DELT)
      DO 1114 J=1,M
 1114 EX(I,J) = SLOPE(I)*X(I,J) + XAV(I)
 1115 CONTINUE
      PRINT 1116, N, M, DELT
 1116 FORMAT(4H1 N=,I2,4H, M=,I4,7H,DELTA=,E14.8/93H0THE EXPERIMENTAL VA
     1LUES OF THE INDEPENDENT VARIABLES WITH THE CORRESPONDING CODED VAL
```

```
    2UES ARE)
     DO 1118 J=1,N
     PRINT 1117, J, XMIN(J), XMAX(J), SLOPE(J), XAV(J)
1117 FORMAT(//33H0FOR INDEPENDENT VARIABLE NUMBER ,I2,46H THE EQUATION
    1FOR THE DESIGN POINTS IS EX=AX+B/7H WHERE //6HEXMIN=E14.7,11H AND
    2EXMAX=E14.7//6H   A=E14.7,11H AND    B=E14.7)
1118 PRINT 1119, (I, X(J,I), EX(J,I), I=1,M)
1119 FORMAT(45H0EXPERIMENT - - - -  X CODED  - - - - EX REAL//(I8,E24.7
    1       ,E18.7))
     PUNCH 201, N, M, DELT
 201 FORMAT (2I5, E12.8)
 202 FORMAT (6E12.7)
 203 FORMAT (8E10.3)
     PUNCH 202, (SLOPE(I),XAV(I), I=1,N)
     DO 22 L=1,M
  22 PUNCH 203, (X(I,L), I=1,N)
     END
```

```
      PROGRAM FIT
      DIMENSION A(45),YY(273),YD(273),X(8,273),P(8,8),W(8),
     1          Z(8),Q(8),ZZ(8,8),QQ(8,8),B(8),BB(8,8)
      COMMON/BLOCK1/Y(273)
      COMMON/BLOCK2/SLOPE(8), XAV(8)
      COMMON/BLOCK3/XMIN(8), XMAX(8)
      COMMON/BLOCK4/BO(1),REALB(8),REALC(8,8)
      COMMON/BLOCK8/EX(8,273)
      COMMON/BLOCK9/XX(36,143),C(45),CK(45),PARTIAL(287,36),KN,ZS,RU,RL
C
 1120 READ 101, N, M, DELT
  101 FORMAT (2I5, E12.8)
      READ 102, (SLOPE(I),XAV(I), I=1,N)
  102 FORMAT (6E12.7)
      DO 2 L=1,M
    2 READ 203, (X(I,L), I=1,N)
      READ 202, (Y(L), L=1,M)
  202 FORMAT (8E10.6)
  203 FORMAT (8E10.3)
C
C**********COMPUTE THE Z AND THE Q**********
C
      N2 = 2**N
      KN = (N+1)*(N+2)/2
      AK=ZO=ZS=0.  $  QO = M  $  DELT4 = 2.*DELT**4
      DO 20 L=1,M
      AK = AK + X(1,L)**2
      ZS = ZS + Y(L)**2  $  ZO = ZO - 2.*Y(L)
   20 CONTINUE
      AK=AK/M
      DO 25 I=1,N  $  Z(I) = 0.0  $  DO 21 L=1,M
   21 Z(I) = Z(I) - 2.*X(I,L)*Y(L)  $  Q(I) = M*AK
      DO 25 J=I,N  $  ZZ(I,J) = 0.0  $  DO 25 L=1,M
      IF(I .EQ. J)22,24
   22 ZZ(I,J) = ZZ(I,J) - 2.*Y(L)*(X(I,L)**2 - AK) $ QQ(I,J) = DELT4
      GO TO 25
   24 ZZ(I,J) = ZZ(I,J) - 2.*Y(L)*X(I,L)*X(J,L)  $  QQ(I,J) = N2
   25 CONTINUE
```

```
C
C
C**********EQUATE Z AND Q TO SINGLE SUBSCRIPT CONSTANTS**********
C
      C(1) = ZO  $ CK(1) = QO
      DO 40 I=1,N
      II=I+1
      C(II) = Z(I)  $  CK(II) = Q(I)
      DO 40 L=1,M
      XX(1,L) = 1.
      XX(II,L) = X(I,L)
   40 CONTINUE
      NM=N-1  $  NP=N+1
      DO 60 I=1,N  $  DO 50 J=I,N
      IJ=J+NP
      C(IJ) = ZZ(I,J)  $  CK(IJ) = QQ(I,J)
      DO 50 L=1,M
      XX(IJ,L) = X(I,L)*X(J,L)
   50 CONTINUE
      NP=NP+NM  $  NM=NM-1
   60 CONTINUE
C
C**********ADJUST SQUARED TERM FOR ORTHOGONALITY
C
      JJ=N+1 $  KJ = JJ + 1
      DO 70 I=KJ,KN,JJ  $  DO 65 L=1,M
   65 XX(I,L) = XX(I,L) - AK
   70 JJ=JJ-1
      PRINT 77
   77 FORMAT (1H1)
      WAY = -1.0
      READ 885, NC, NX, RU, RL
  885 FORMAT (2I5, 2E12.8)
      LO = NX
      M2 = 2*M + 1
      CALL DIRECT (A, WAY, KN, M2, M)
```

```
C**********EQUATE COEFFICIENTS FROM QUADRATIC PROGRAM TO STANDARD FORM
C          FOR USE IN SUBR. REAL IN ORDER TO USE REAL VALUES OF X(I)

      AZRO = A(1)
      DO 35 I=1,N
      II = I + 1
   35 B(I) = A(II)
      IJ = N + 2
      DO 37 I=1,N  $  DO 37 J=I,N
      BB(I,J) = A(IJ)
   37 IJ = IJ + 1
      BZRO = AZRO  $  DO 38 I=1,N
   38 BZRO = BZRO - AK*BB(I,I)
C
C**********COMPUTE Y USING CODED X**********
C
      DO 31 L=1,M $ SUM = 0. $ DO 30 I=1,N $ SUM = SUM + B(I)*X(I,L)
      DO 29 J=I,N
   29 SUM = SUM + BB(I,J)*X(I,L)*X(J,L)
   30 CONTINUE $ YY(L) = BZRO + SUM
   31 CONTINUE
C
C**********COMPUTE STATISTICS**********
C
      G = SYSQ = 0.0
   32 DO 33 L=1,M $ YD(L) = Y(L) - YY(L)
      G = G + (YD(L)*YD(L))
   33 SYSQ = SYSQ + (Y(L)*Y(L))
      RSQ = 1. - G/SYSQ
      EV=G/(M-KN)
      SIGMA = SQRTF(EV)
C
C**********OUTPUT**********
C
      PRINT 100, (NC, N, SIGMA, M, RSQ, KN, EV, DELT, G, N2,  (
     1L, X(1,L), X(2,L), X(3,L), X(4,L), X(5,L), X(6,L), X(7,L), X(8,L),
     2Y(L), YY(L), YD(L), L=1,M))
```

```
  100 FORMAT(1H1, 61X, 12H CASE NUMBER, I6//8X, 22H NUMBER OF VARIABLES
     1=, I4, 10X, 21H STANDARD DEVIATION =, E12.5//8X,                26H
     2NUMBER OF DESIGN POINTS =,I5,5X,18H GOODNESS OF FIT =,E12.5//8X,25
     3H NUMBER OF COEFFICIENTS =,  I6,5X,17H ERROR VARIANCE =,E12.5//8X,
     47H DELT = , F9.6, 20X, 4H G =, E12.5//8X, 22H THE NUMBERS AFTER L
     5=, I3, 24H HAVE THE VALUE OF DELT //8X,78H  L    X1  X2  X3  X4  X
     65  X6  X7  X8   INPUT VALUE   COMP. VALUE   DIFFERENCE   /
     7 (I11, F6.0, 7F4.0, 3E14.5))
C
 1000 CALL REAL (N, BZRO, B, BB, M, NC)
  304 IF(NX .EQ. 0)1200,114
 1200 STOP
C
C**********EXTRA POINTS COMPUTATION**********
C
  114 DO 620 L=1,LO $ SUM = 0. $ DO 619 I=1,N
      READ 613, P(I,L)
  613 FORMAT (8E10.6)
      SUM = SUM + REALB(I)*P(I,L)
      DO 618 J=I,N
  618 SUM = SUM + REALC(I,J)*P(I,L)*P(J,L)
  619 CONTINUE  $  W(L) = BO + SUM
  620 CONTINUE  $  I=N+1
  622 DO 623 L=1,LO
  623 P(I,L) = 0.0 $ I = I + 1
      IF(I -  8) 622,622,624
  624 PRINT 625, (NC,(P(1,L), P(2,L), P(3,L), P(4,L), P(5,L), P(6,L), P(
     17,L), P(8,L), W(L), L=1,LO))
  625 FORMAT(40X,12H CASE NUMBER, I6,//5X,45H X1    X2    X3    X4    X5
     1    X6    X7    X8,7X,2H Y,//(0PF9.2,0P7F6.2,1PE15.6))
  626 GO TO 1200
      END

      SUBROUTINE REAL (N, BZRO, B, C, M, NC)
C     DETERMINE COEFFICIENTS FOR REAL X
C
      DIMENSION B(8), C(8,8), REALC(8,8), SA(8), SB(8), RATIO(8),
```

```
 1           REALB(8)
  COMMON/BLOCK2/SA,SB
  COMMON/BLOCK4/BO(1),REALB,REALC
  BO = BZRO $ DO 1 I=1,N $ REALB(I) = 0. $ RATIO(I) = SB(I)/SA(I)
  DO 1 J=1,N $ REALC(I,J) = 0. $  C(J,I) = C(I,J)
1 CONTINUE
  DO 2 I=1,N $ BO = BO - B(I)*RATIO(I)
  REALB(I) = (B(I) - 2.*C(I,I)*RATIO(I))/SA(I)
  DO 2 J=I,N $ BO = BO + C(I,J)*RATIO(I)*RATIO(J)
2 REALC(I,J) = REALC(I,J) + C(I,J)/(SA(I)*SA(J))
  DO 4 I=1,N $ DO 4 J=1,N
  IF(I .EQ. J)4,3
3 REALB(I) = REALB(I) - C(I,J)*RATIO(J)/SA(I)
4 CONTINUE
  PRINT 5, NC
5 FORMAT (27H1VALUES OF THE COEFFICIENTS,13X,11HCASE NUMBER,I6/
 1         12H0FOR CODED X,28X,10HFOR REAL X/4H I J,36X,3HI J)
  J = 0 $ I = 0
  PRINT 6, I, J, BZRO, I, J, BO
  PRINT 6,(I, J, B(I), I, J, REALB(I), I=1,N)
  PRINT 6,((I, J, C(I,J), I, J, REALC(I,J),J=I,N),I=1,N)
6 FORMAT(     2I2, E20.8,15X,2I2,E20.8)
  RETURN
  END

  SUBROUTINE DIRECT (A, WAY, K, N, M)
  DIMENSION A(45)
  COMMON/BLOCK5/FCHG,PERC,LX,LY,MQ,TEST,KFREQ,MK,KFREQS
  COMMON/BLOCK6/BL(45),UL(45),FNN,IFAIL,ISTOP,IFCHG,NUMEX,NOPTN
  COMMON/BLOCK7/PER(45),DEL(45),ALO(45),ALN(45),SUD(45),FAIL(45),
 1              FOLD,FN,MM,IPARA1,IPARA2,DELP1,DELP2
  COMMON/BLOCK8/FNEG(274),ILATE(274),KCON,ICINE,INE,EP(45),Y(45),
 1              VECTOR(274),CINE(549),F(549),DUMMY( 171)
  IF(K.EQ.0)44,45
44 STOP
45 CONTINUE
  READ 972
```

```
      PRINT 972
  972 FORMAT ( 80H
     1                                                     )
   90 CONTINUE
      READ 2, MQ, PERC
    2 FORMAT (I5,F14.8)
      PRINT 105,PERC
      DO 20 I=1,K
   20 PER(I) = PERC
  105 FORMAT (/33H THE INITIAL ALLOWABLE CHANGE IS ,F14.8, 4H * AI9    )
      MK = 0
      NOPTN = 0
      NUMEX = 0
      MM = 0
    5 READ 3,(A(I),I=1,K)
      READ 3, (UL(I),I=1,K)
      READ 3,(BL(I),I=1,K)
      READ 6, TEST,LX,LY
      READ 6, FCHG,KFREQS,KFREQ
    3 FORMAT (8E10.6)
    6 FORMAT (E10.6, 2I5)
      IFAIL = 2
      IFCHG = 2
      KCON = 1
      ISTOP = 2
      CALL SUMS (A, K, N, M)
      GO TO (1000,1001),ISTOP
 1000 PRINT 1002
 1002 FORMAT (/42H INITIAL GUESSES NOT IN REGION OF SEARCH   )
      IFAIL = 1
      FNN = 0.0
      DO 600 J=1,INE
      FNN = FNN + FNEG(J)
  600 CONTINUE
      WAYO = WAY
      WAY = 1.0
 1001 FN=FNN
```

```
      FOLD = FN
 1102 CONTINUE
      IF (FOLD )501,509,501
  509 FOLD = (FCHG)**2
  501 CONTINUE
      PRINT 51,(A(I),I=1,K)
   51 FORMAT (33H THE INITIAL VALUES OF A(I) ARE      / 8E14.6)
      PRINT 97,FN
   97 FORMAT (26H THE INITIAL VALUE OF FN = E15.8)
      DO 8 I=1,K
      SUD(I) = FAIL(I) = 0.0
      ALO(I)=A(I)
    8 ALN(I)=A(I)
      CALL PATSER (A, WAY, K, M, N)
      GO TO (800,810),IFAIL
  800 CONTINUE
      GO TO (850,815),IFCHG
  850 CONTINUE
      PRINT 1850
 1850 FORMAT (/30H HAVE FOUND FEASIBLE POINT        )
      IFAIL = 2
      IFCHG = 2
      WAY = WAYO
      MK = 0
      NOPTN = 0
      NUMEX = 0
      MM = 0
      ISTOP = 2
      DO 250 J=1,K
      PER(J) = PERC
  250 CONTINUE
      GO TO 1001
  815 CONTINUE
  810 CONTINUE
 1005 IF ( NOPTN -MQ ) 40,41,41
   41 CONTINUE
   91 PRINT 50
```

```
   50 FORMAT (28H PROBLEM FAILED TO CONVERGE.       )
      PRINT 82,NUMEX,NOPTN
   82 FORMAT (30H NUMBER EXPLORATORY SEARCHES =    I5// 39H NUMBER OF SUC
     1CESSFULL PATTERN MOVES = I5/)
      CALL OUTPUT (A, K)
      DO 111 I=1,K
  111 A(I) = ALN(I)
      RETURN
   40 PRINT 60
   60 FORMAT (///  29H **** THESE ARE FINAL ANSWERS / )
      PRINT 82,NUMEX,NOPTN
      CALL OUTPUT (A, K)
      DO 110 I=1,K
  110 A(I) = ALN(I)
      PRINT 1999
 1999 FORMAT (1H1 )
      RETURN
      END

      SUBROUTINE PATSER (A, WAY, K, M, N)
      DIMENSION A(45)
      COMMON/BLOCK5/FCHG,PERC,LX,LY,MQ,TEST,KFREQ,MK,KFREQS
      COMMON/BLOCK6/BL(45),UL(45),FNN,IFAIL,ISTOP,IFCHG,NUMEX,NOPTN
      COMMON/BLOCK7/PER(45),DEL(45),ALO(45),ALN(45),SUD(45),FAIL(45),
     1              FOLD,FN,MM,IPARA1,IPARA2,DELP1,DELP2
      COMMON/BLOCK8/FNEG(274),ILATE(274),KCON,ICINE,INE,EP(45),Y(45),
     1              VECTOR(274),CINE(549),F(549),DUMMY( 171)
      COMMON/BLOCK9/XX(36,143),C(45),CK(45),PARTIAL(287,36),KN,ZS,RU,RL
      NAC = 1
      KDAPT = 0
      IACCEL = 0
      ITED = 0
      ITER =0
      MK = 0
      IADAPT = 1
      JJ = 0
```

```
C***** SECTION A *****

   90 CONTINUE
      DO 6 I = 1,K
      DEL (I) = PER(I)*ALO(I)
      IF ( DEL(I) - PERC )1006,1007,1007
 1006 CONTINUE
      DEL(I) = PERC
 1007 CONTINUE
    6 CONTINUE
      INK =0
      ITER = 0
      NOMEX = 0
      GO TO 301

C***** SECTION B *****

  301 CONTINUE
   30 CONTINUE
      FN = FOLD
      DO 1 I = 1,K
    1 A(I) = ALN(I)

C***** SECTION C *****

  313 CONTINUE
      CALL EXPSER (A, WAY, K, N)

C***** SECTION D, EXPSER IS TYPE I *****

      GO TO (800,810),IFAIL
  800 CONTINUE
      GO TO (850,815),IFCHG
  815 CONTINUE
  810 CONTINUE
  100 IF(ABSF((FOLD-FN)/FOLD) - FCHG) 85,85,11
   11 CONTINUE
```

```
      IF((FOLD -FN ) * WAY) 33,85,85

C***** SECTION E *****
C***** SECTION F, EXPSER IS TYPE I *****

   33 DO 2 I= 1,K
C     - - - - - - - THIS PATH MEANS A SUCCESS - - - - - -
      ALO (I) = ALN(I)
    2 ALN(I) = A(I)
      NAC = 1
      IACCEL = 0
      IADAPT = 1
      FOLD = FN
      IF (FOLD )501,500,501
  500 FOLD = (FCHG)**2
  501 CONTINUE
C     - - - - - - - START PATTERN MOVE - - - - - -
      DO 3 I=1,K
      CH = 2.*A(I) - ALO(I)
      IF(CH - BL(I))2010,2010,2011
 2010 A(I) = BL(I)
      GO TO 2020
 2011 IF(UL(I) - CH)2012,2012,2013
 2012 A(I) = UL(I)
      GO TO 2020
 2013 A(I) = CH
 2020 CONTINUE
      CALL SUMS (A, K, N, M)
      GO TO (801,811),IFAIL
  801 CONTINUE
      GO TO (850,816),IFCHG
  816 CONTINUE
      GO TO 101
  811 CONTINUE
      GO TO (101,102),ISTOP
  101 A(I) = ALN(I)
      GO TO 3
```

```
  102 CONTINUE
  103 FN = FNN
    3 CONTINUE
      CALL EXPSER (A, WAY, K, N)

C***** SECTION D, EXPSER IS TYPE II *****

      GO TO (802,812),IFAIL
  802 CONTINUE
      GO TO (850,817),IFCHG
  817 CONTINUE
  812 CONTINUE
  106 IF(ABSF((FOLD-FN)/FOLD) - FCHG) 53,53,10
   10 CONTINUE
      IF (( FOLD - FN) *WAY) 78,53,53

C***** SECTION F, EXPSER IS TYPE II *****

   78 CONTINUE
      NOPTN = NOPTN + 1
C     - - - - - - PATTERN MOVE A SUCCESS - - - - - -
      GO TO (56,308),LY
  308 INK = INK+1
      IF(INK - KFREQS)56,57,56
   57 CONTINUE
      PRINT 58,NOPTN
   58 FORMAT (20X,15H THIS IS AFTER I5,56H TYPE II EXP. SEARCH, THE PATT
     1ERN MOVE WAS A SUCCESS.                          //)
      CALL OUTPUT (A, K)
      INK = 0
   56 CONTINUE
      IF ( NOPTN -MQ   ) 33,33,40

C***** SECTION G, EXPSER IS TYPE II *****

   53 CONTINUE
      GO TO (301,80),LX
```

```
   80 ITER = ITER +1
      MK = MK + 1
      IF( ITER - KFREQ) 301,300,301
  300 CONTINUE
      ITER =0
      PRINT 91,MK
   91 FORMAT ( 20X,10H HAVE HAD I3,24H PATTERN MOVE FAILURES.          //)
      CALL OUTPUT (A, K)
      GO TO 301

C***** SECTION G, EXPSER IS TYPE I *****

   85 CONTINUE
      NOMEX = NOMEX + 1
      MM= MM+1
      DO 26 I = 1,K
      IF(PER(I) - TEST) 26,26,27
   26 CONTINUE
      GO TO 40
   27 CONTINUE
      GO TO (302,81),LX
   81 CONTINUE
      ITED = ITED + 1
      IF(ITED - KFREQ)302,303,302
  303 CONTINUE
      ITED = 0
      PRINT 93,NOMEX
   93 FORMAT (// 20X,10H HAVE HAD  I3,40H TYPE I EXPLORATORY SEARCH FAIL
     1URES.                          //)
      CALL OUTPUT (A, K)
  302 CONTINUE
      GO TO (803,813),IFAIL
  803 CONTINUE
      GO TO (850,818),IFCHG
  818 CONTINUE
      GO TO 450
  813 CONTINUE
```

```
      GO TO ( 400,450 ), IADAPT
  400 GO TO ( 425,450 ),ICINE
  425 CONTINUE
C     - - - - - - START ADAPTIVE MOVE - - - - - -
      KDAPT = KDAPT + 1
      LMN = 0
      IPAT = 1
      DO 445 J=1,K
      Y(J) = A(J)
  445 CONTINUE

C***** SECTION H *****

  426 CONTINUE
      CALL GRAD (A, K, N, M)

C***** SECTION J *****

  455 CONTINUE
      DO 475 J=1,K
      VECTOR(J) = 0.0
      DO 475 I=1,KCON
      VECTOR(J) = VECTOR(J) + PARTIAL(ILATE( I  ),J)
  475 CONTINUE
      DENOM = 0.0
      DO 476 J=1,K
      DENOM = DENOM + VECTOR(J)**2
  476 CONTINUE
      DENOM = SQRTF(DENOM)
      DO 477 J=1,K
      VECTOR(J) = VECTOR(J)/DENOM
  477 CONTINUE
  401 DO 402 J=1,K
      EP ( J ) =    VECTOR(J)         + PARTIAL ( 1,J ) * WAY
  402 CONTINUE
      DENOM = 0.0
      DO 440 J=1,K
```

```
      DENOM = DENOM + EP(J)**2
  440 CONTINUE
      DENOM = SQRTF( DENOM )
      IF ( DENOM - FCHG )442,442,443
  443 CONTINUE
      DO 441 J=1,K
      EP(J) = EP(J)/DENOM
      EP(J) = EP(J)*PER(J)
  441 CONTINUE
  427 CONTINUE
      DO 428 J=1,K
      CH = A(J) + EP(J)*(1 + IACCEL)
      IF(CH - BL(J))7780,7780,7781
 7780 A(J) = BL(J)
      GO TO 428
 7781 IF(UL(J) - CH)7782,7782,7783
 7782 A(J) = UL(J)
      GO TO 428
 7783 A(J) = CH
  428 CONTINUE
      KCON = KCON + 1
      GO TO 408

C***** SECTION I *****

  442 CONTINUE
      PRINT 1442
 1442 FORMAT(51H **** OBJECTIVE AND RESTRAINT ARE NEARLY PARALLEL     /)
      GO TO (460,461,462),IPAT
  460 CH = A(IPARA1) + DELP1
      IPAT = 2
      IQP = IPARA1
      GO TO 2030
  461 CH = A(IPARA2) + DELP2
      IPAT = 3
      IQP = IPARA2
 2030 IF(CH - BL(IQP))2031,2031,2032
```

```
 2031 A(IQP) = BL(IQP)
      GO TO 426
 2032 IF(UL(IQP) - CH)2033,2033,2034
 2033 A(IQP) = UL(IQP)
      GO TO 426
 2034 A(IQP) = CH
      GO TO 426
  462 CONTINUE
      STOP

C***** SECTION K *****

  408 CALL SUMS (A, K, N, M)
      FN = FNN
      GO TO (419,403),ISTOP
  419 CONTINUE
      GO TO 411
  411 CONTINUE
      DO 478 J=1,K
      A(J) = Y(J)
  478 CONTINUE
      IF ( ILATE(1) - ILATE(2) )2001,409,2001
 2001 CONTINUE
      PRINT 2000,(ILATE(I),I=1,KCON)
 2000 FORMAT (10X,22I5)
      IF (KCON - 7*K)457,457,415
  457 CONTINUE
      GO TO 455
  403 CONTINUE
      IF ((FOLD - FN )*WAY)404,409,409
  404 CONTINUE
      GO TO(470,471),LX
  471 CONTINUE
      IF(KDAPT - KFREQ)470,472,470
  472 CONTINUE
      KDAPT = 0
      PRINT 1404
```

```
 1404 FORMAT (/30H ADAPTIVE MOVE SUCCESSFUL                    /)
      CALL OUTPUT (A, K)
  470 CONTINUE
      MM= 0
      KCON = 1
      IACCEL = IACCEL + 1
      NAC = 2
      DO 405 J=1,K
      ALO(J) = Y(J)
      ALN(J) = A(J)
  405 CONTINUE
      FOLD = FN
      GO TO 313

C***** SECTION L *****

  409 CONTINUE
      DO 413 J=1,K
      IF ( ABSF( EP(J)) - TEST)415,415,413
  413 CONTINUE
      NAC = 1
      IACCEL = 0
      LMN = LMN + 1
      IF (LMN - 1)487,487,488
  487 CONTINUE
  488 CONTINUE
      FLMN = LMN
      DO 410 J=1,K
      EP(J) = EP(J)/EXPF( FLMN )
      CH = Y(J) + EP(J)
      IF(CH - BL(J))7790,7790,7791
 7790 A(J) = BL(J)
      GO TO 410
 7791 IF(UL(J) - CH)7792,7792,7793
 7792 A(J) = UL(J)
      GO TO 410
 7793 A(J) = CH
```

```
  410 CONTINUE
      GO TO 408

C***** SECTION M *****

  415 CONTINUE
      GO TO(480,481),LX
  481 CONTINUE
      IF(KDAPT - KFREQ)480,482,480
  482 CONTINUE
      KDAPT = 0
      PRINT 1415
 1415 FORMAT (/30H ADAPTIVE MOVE FAILED                  /)
      CALL OUTPUT (A, K)
  480 CONTINUE
      KCON = 1
      DO 451 J=1,K
      A(J) = Y(J)
  451 CONTINUE
      IADAPT = 2
      GO TO 450

C***** SECTION N *****

  450 CONTINUE $  FQF = MM
      DO 22 I=1,K
   23 PER(I) = PER(I) / EXPF(FQF)
   22 CONTINUE
      DO 7 I =1,K
      FAIL (I) = 0.0
    7 DEL(I) = PER(I) * ALO(I)
      GO TO 30
  850 CONTINUE
   40 CONTINUE
      END

      SUBROUTINE EXPSER (A, WAY, K, N)
```

```
      DIMENSION AA(45), A(45)
      COMMON/BLOCK6/BL(45),UL(45),FNN,IFAIL,ISTOP,IFCHG,NUMEX,NOPTN
      COMMON/BLOCK7/PER(45),DEL(45),ALO(45),ALN(45),SUD(45),FAIL(45),
     1              FOLD,FN,MM,IPARA1,IPARA2,DELP1,DELP2
      COMMON/BLOCK8/FNEG(274),ILATE(274),KCON,ICINE,INE,EP(45),Y(45),
     1              VECTOR(274),CINE(549),F(549),DUMMY( 171)
      IP = 1
      ICINE = 2
      DO 20 I=1,K
   20 AA(I) = A(I)
      NUMEX = NUMEX +1
    9 NI=1
   10 I=1

C***** SECTION A *****

      IF(NI-K)11,11,30
   11 CONTINUE
      CH = A(NI) + DEL(NI)
      IF(CH - BL(NI))7760,7760,7761
 7760 A(NI) = BL(NI)
      GO TO 21
 7761 IF(UL(NI) - CH)7762,7762,7763
 7762 A(NI) = UL(NI)
      GO TO 21
 7763 A(NI) = CH

C***** SECTION B *****

   21 CONTINUE
      CALL SUMS (A, K, N, M)
      GO TO (800,810),IFAIL
  800 CONTINUE
      GO TO (850,815),IFCHG
  815 CONTINUE
      GO TO 100
  810 CONTINUE
```

```
C***** SECTION C *****

      GO TO (13,100),ISTOP
  100 CONTINUE

C***** SECTION D *****

      IF((FN-FNN)*WAY)12,23,23
   23 CONTINUE
      GO TO (13,26),IFAIL
   26 CONTINUE

C***** SECTION F *****

      GO TO (24,25),IP
   24 CONTINUE
      IPARA1 = NI
      IPARA2 = NI
      DELP1 = DEL(NI)
      DELP2 = DEL(NI)
      IP = 2
      GO TO 13
   25 CONTINUE
      IPARA2 = IPARA1
      IPARA1 = NI
      DELP2 = DEL(IPARA1)
      DELP1 = DEL(NI)
      GO TO 13
   12 SUD(NI)=SUD(NI)+1.0
      FN=FNN
      NI=NI+1
      MM=0
      GO TO 10

C***** SECTION E *****
```

```
   13 GO TO (14,15),I
   14 CH = A(NI) - 2.*DEL(NI)
      IF(CH - BL(NI))7770,7770,7771
 7770 A(NI) = BL(NI)
      GO TO 7774
 7771 IF(UL(NI) - CH)7772,7772,7773
 7772 A(NI) = UL(NI)
      GO TO 7774
 7773 A(NI) = CH
 7774 CONTINUE
      DEL (NI) = - DEL(NI)
      SUD(NI) = 0.0
      I=2
      GO TO 21
   15 FAIL(NI)=FAIL(NI)+1.0
      SUD(NI)=0.0
      A(NI) = AA(NI)
      NI=NI+1
      GO TO 10
  850 CONTINUE
   30 CONTINUE
      END

      SUBROUTINE SUMS (A, K, N, M)
      DIMENSION A(45)
      COMMON/BLOCK6/BL(45),UL(45),FNN,IFAIL,ISTOP,IFCHG,NUMEX,NOPTN
      COMMON/BLOCK8/FNEG(274),ILATE(274),KCON,ICINE,INE,EP(45),Y(45),
     1              VECTOR(274),CINE(549),F(549),DUMMY( 171)
      COMMON/BLOCK9/XX(36,143),C(45),CK(45),PARTIAL(287,36),KN,ZS,RU,RL
      F(1) = ZS
      DO 75 I=1,KN
   75 F(1) = F(1) + C(I)*A(I) + CK(I)*A(I)**2
      RDIFF = RU - RL
      DO 80 L=1,M $  LP1 = L + 1
      F(LP1) = -RL
      DO 80 I=1,KN
      F(LP1) = F(LP1) + A(I)*XX(I,L)
```

```
80 CONTINUE
   DO 85 L=1,M  $  LP1 = L + 1  $  LP1M = LP1 + M
85 F(LP1M) = -F(LP1) + RDIFF
   FNN = F(1)
   ISTOP = 2
   L = 1
   DO 6 I=2,N
 4 IF(F(I))5,6,6
 5 CONTINUE
   L = L + 1
   ICINE = 1
   ISTOP = 1
   FNEG(L-1) = F(I)
   IF ( L - 2 )8,8,12
12 CONTINUE
   IF ( CINE(ILATE(KCON)) - F(I))6,6,8
 8 CONTINUE
   CINE(I) = F(I)
   ILATE(KCON) = I
 6 CONTINUE
   INE = L-1
   GO TO (30,31),IFAIL
30 CONTINUE
   IF (INE)33,33,34
34 CONTINUE
   FNN = 0.0
   DO 32 J=1,INE
   FNN = FNN + FNEG(J)
32 CONTINUE
31 CONTINUE
   GO TO 7
33 CONTINUE
   IFCHG = 1
 7 RETURN
   END

   SUBROUTINE GRAD (A, K, N, M)
```

```
      DIMENSION A(45)
      COMMON/BLOCK9/XX(36,143),C(45),CK(45),PARTIAL(287,36),KN,ZS,RU,RL
      DO 84 I=1,KN
      PARTIAL(1,I) = C(I) + 2.*CK(I)
      DO 84 L=1,M
      LP1 = L+1
      PARTIAL(LP1,I) = XX(I,L)
      LP1M = LP1+M
      PARTIAL(LP1M,I) = -PARTIAL(LP1,I)
   84 CONTINUE
      DO 4 I = 1,N
      DENOM = 0.0
      DO 3 J = 1,K
      DENOM = DENOM + PARTIAL ( I,J )**2
    3 CONTINUE
      DENOM = SQRTF ( DENOM )
      DO 4 J = 1,K
      PARTIAL (I,J ) = PARTIAL ( I,J )/DENOM
    4 CONTINUE
      RETURN
      END

      SUBROUTINE OUTPUT (A, K)
      DIMENSION A(45)
      COMMON/BLOCK7/PER(45),DEL(45),ALO(45),ALN(45),SUD(45),FAIL(45),
     1              FOLD,FN,MM,IPARA1,IPARA2,DELP1,DELP2
      PRINT 90,FOLD,FN
   90 FORMAT(54H THE CURRENT BASE VALUE OF THE FUNTION F(ALN(I)) IS =
     1E20.8 // 34H THE FUNCTION TESTED F(A(I)) IS =       E20.10   //)
      PRINT 3
      DO 2 I=1,K
    2  PRINT 4,I,ALO(I) ,ALN(I), A(I), PER(I),DEL(I)
    4 FORMAT(I3,5E23.10)
    3 FORMAT(3H AI,7X,14H PREVIOUS BASE 10X,13H CURRENT BASE11X,10H TEST
     1 BASE12X,15H PERCENT CHANGE12X,6H DELTA //)
      END
```

APPENDIX C

SAMPLE COMPUTER PRINTOUT FOR CASE ONE

```
N= 2,M= 9,DELTA=1.00000000E 00

THE EXPERIMENTAL VALUES OF THE INDEPENDENT VARIABLES WITH THE
CORRESPONDING CODED VALUES ARE

FOR INDEPENDENT VARIABLE NUMBER  1 THE EQUATION FOR THE
DESIGN POINTS IS EX=AX+B, WHERE

EXMIN=                 0 AND EXMAX= 2.0000000E 00

    A= 1.0000000E 00 AND     B= 1.0000000E 00


EXPERIMENT - - - -  X CODED  - - - - EX REAL

       1          -1.0000000E 00                 0
       2           1.0000000E 00     2.0000000E 00
       3          -1.0000000E 00                 0
       4           1.0000000E 00     2.0000000E 00
       5          -1.0000000E 00                 0
       6                        0    1.0000000E 00
       7           1.0000000E 00     2.0000000E 00
       8                        0    1.0000000E 00
       9                        0    1.0000000E 00


FOR INDEPENDENT VARIABLE NUMBER  2 THE EQUATION FOR THE
DESIGN POINTS IS EX=AX+B, WHERE

EXMIN=                 0 AND EXMAX= 2.0000000E 00

    A= 1.0000000E 00 AND     B= 1.0000000E 00


EXPERIMENT - - - -  X CODED  - - - - EX REAL

       1          -1.0000000E 00                 0
       2          -1.0000000E 00                 0
       3           1.0000000E 00     2.0000000E 00
       4           1.0000000E 00     2.0000000E 00
       5                        0    1.0000000E 00
       6          -1.0000000E 00                 0
       7                        0    1.0000000E 00
       8           1.0000000E 00     2.0000000E 00
       9                        0    1.0000000E 00
```

QUAD L.S. FIT Y= 4*X(1)*X(2) - 2*X(1)**2 - X(2)**4
STANDARD DEVIA = .5   UPPER BOUND = +1.0

THE INITIAL ALLOWABLE CHANGE IS   .20000000 * A
THE INITIAL VALUES OF A(I) ARE
0 0 0 0 0 0
THE INITIAL VALUE OF FN = 5.03613441E 02

***** THESE ARE FINAL ANSWERS

NUMBER EXPLORATORY SEARCHES = 21

NUMBER OF SUCCESSFULL PATTERN MOVES = 8

THE CURRENT BASE VALUE OF THE FUNCTION F(ALN(I)) IS 4.76756204E-01

THE FUNCTION TESTED F(A(I)) IS = 4.7675620391E-01

AI	PREVIOUS BASE	CURRENT BASE	TEST BASE
1	-5.000000000E 00	-5.000000000E 00	-5.000000000E 00
2	-1.999999999E-01	-1.852848223E-01	-1.852848223E-01
3	-4.200000000E 00	-4.200000000E 00	-4.200000000E 00
4	-2.000000000E 00	-2.000000000E 00	-2.000000000E 00
5	3.800000000E 00	3.800000000E 00	3.800000000E 00
6	-6.799999999E 00	-7.256170506E 00	-7.256170506E 00

NUMBER OF VARIABLES = 2   STANDARD DEVIATION = 3.98646E-01

NUMBER OF DESIGN POINTS = 9   GOODNESS OF FIT = 9.99053E-01

NUMBER OF COEFFICIENTS = 6   ERROR VARIANCE = 1.58919E-01

DELT = 1.000000   G = 4.76756E-01

THE NUMBERS AFTER L = 4 HAVE THE VALUE OF DELT

L	X1	X2	X3	X4	X5	X6	X7	X8	INPUT VALUE	COMP. VALUE	DIFFERENCE
1	-1	-1	0	0	0	0	0	0	-3.38294E-03	9.98947E-02	-1.03278E-01
2	1	-1	0	0	0	0	0	0	-8.02706E 00	-7.87067E 00	-1.56388E-01
3	-1	1	0	0	0	0	0	0	-1.61488E 01	-1.59001E 01	-2.48743E-01
4	1	1	0	0	0	0	0	0	-8.69688E 00	-8.67067E 00	-2.62066E-02
5	-1	0	0	0	0	0	0	0	-5.16249E-01	-6.43935E-01	1.27685E-01
6	0	-1	0	0	0	0	0	0	-1.89853E 00	-1.88539E 00	-1.31425E-02
7	1	0	0	0	0	0	0	0	-7.86944E-01	-1.01450E 00	2.27561E-01
8	0	1	0	0	0	0	0	0	-9.88182E 00	-1.02854E 01	4.03570E-01
9	0	0	0	0	0	0	0	0	7.86146E-01	1.17078E 00	-3.84635E-01

VALUES OF THE COEFFICIENTS

FOR CODED X

I	J	
0	0	1.17078034E 00
1	0	-1.85284822E-01
2	0	-4.20000000E 00
1	1	-2.00000000E 00
1	2	3.80000000E 00
2	2	-7.25617051E 00

FOR REAL X

I	J	
0	0	9.98946537E-02
1	0	1.47151771E-02
2	0	6.51234101E 00
1	1	-2.00000000E 00
1	2	3.80000000E 00
2	2	-7.25617051E 00